Lab 4
“while” loops and “for” loops

The purpose of this lab is to

• introduce you to the concept of a for loop,
• gain experience distinguishing between a “while” loop (which is a more general type of loop than a “for” loop) and a “for” loop, and
• gain more practice writing loops.

Technical Section

A. “while” Loops or “for” loops

In the last lab, we practiced using a counting loop, which (as shown below) features 4 sections:

```java
int i=1;
while (i<=6)
{
  System.out.println("Try Again");
  i++;
}
```

• **initialization** (statements executed once at the beginning of the loop to initialize variables in the conditional),
• **conditional** (expression evaluated at the beginning of every loop to indicate whether to loop over the body or quit looping)
• **update** (statements executed after every iteration over the body to update variables in the conditional), and
• **body** (statements to repeat).

One could argue that the “update” component is also part of the body. Here we distinguish the “update” and “body” by focusing on the part of the loop that one is trying to repeat (processing another user input, another record from a database, another element from a table...) versus a statement that is creating the infrastructure for repeating.

**Label each part of the while loop above as one of the four components. (see the completion sheet to record your response)**

Here is the general format of a “for” loop:

```java
1 for ( initialization ; conditional ; update ) {
2      body
3  }
```

And the same general format for a while loop

```java
1    initialization ;
2    while ( conditional ) {
3      body with update
4  }
```

The following code segment is an example of a for loop that implements the while loop above:

```java
for (int i=1; i<=6; i++)
{ System.out.println("Try Again");
}
```

Can you see the four components in both loops? A for loop was intended as a more convenient way for the programmer to write a count-controlled loop.

Although “while” loops are similar in purpose to “for” loops (both iterate over the statements in the body while the conditional holds true), “while” loops are actually more general and can be used in places where a “for” loop would not typically be used. There is some disagreement among programmers as to which loop should be used in any given instance, but the decision is really not that difficult.

1. Whenever you are executing a “body” of statements a **known** number of times (known when the program reached the beginning of the loop), use a “for” to implement a count-controlled loop.
2. Whenever you are executing a “body” of statements an **unknown** number of times (when the loop starts, you can’t predict how many times it will execute: until you find the end of the file, until the program finds the greatest common denominator,…) use “while”.

Although in terms of performance, “for” and “while” may not differ significantly, they do in terms of readability: whereas “for” loops are more adequate for loops that increment from a lower to an upper bound, “while” loops are better for conditions that can not be anticipated, for example, as a result of evaluating input from the user, or as a result of computations.

In the following exercises we will practice both of these approaches.

---

**Programming Section**

**Exercise 1: Counting Until Divisible by 6 or 17**

In this first exercise you will create a program that uses “Scanner” to read numbers from the user until the last number read is evenly divisible by 3 or by 7.

*If you are not sure whether to use a “for” or “while”, ask your instructor.*

Using jEdit, create a “DivisibleBy6or17.java” file in the “lab04” folder (create this folder if you haven’t already done so), and write a “DivisibleBy6or17” class with a “main” method in it. This method should have a “while” loop that asks repeatedly for a number until a number divisible by 6 or 17 is given. When that occurs, your program should display the number of numbers read, as illustrated in the figure below. Note that the last number should not be counted.

![Command Prompt with program output](image)

**Testing**

Test that your program calculates areas correctly by using the following values

- input = 9474; numbers read = 0
- input = 23, 85; numbers read = 1
- input = 110, 71, 1765, 8765, 888; numbers read = 4
- input = 1, 4, 5, 8, 10, 11, 13, 16, 34; numbers read = 8
When you are finished, submit your program to Web-CAT.
Exercise 2: Finding the Average

This next exercise is similar to the previous one in that you will write a program that reads “n” numbers from the user and then displays the average of all of the number that are divisible by 6 or 17. For example, the figure below shows the case in which the user enters 7 values and the average only includes. If none of the values entered are divisible by 6 or 17, output “NONE DIVISIBLE”.

If you are not sure whether to use a “for” or “while”, ask your instructor.

Using jEdit, create a “FunnyAverage.java” file in the “lab04” folder, and write a “FunnyAverage” class with a “main” method in it. This method should begin by asking the number of integers to read, and will then read as many numbers as requested summing only the numbers divisible by 6 or 17. At the end, the program should display the average of those numbers or “NONE DIVISIBLE”.

Testing

Test that your program calculates areas correctly by using the following values

• numbers to read = 2; input = 3, 4; NONE DIVISIBLE
• numbers to read = 3; input = 6, 17, 12; sum = 35; average = 11.666666666666666
• numbers to read = 4; input = 110, 888, 34, 16; sum = 922; average = 461.0

When you are finished, submit your program to Web-CAT.
Exercise 3: Euclid’s GCD

In this next exercise we will forgo inputs from the user and focus on a particular algorithm that is well geared to practice “while” loops: Euclid’s greater common divisor (GCD) algorithm.

Euclid of Alexandria, who was a Greek mathematician, wrote this algorithm to find the GCD of 2 numbers, that is, the highest number that can evenly divide these numbers without a remainder. The algorithm goes as follows:

Given two positive numbers “a” and “b”, where “a” is greater or equal than “b”, for as long as “b” is greater than 0, calculate the positive difference between “a” and “b” and assign this difference as the new value of either “a” (if “a” is greater than “b”) or “b” (otherwise). At the end of this loop, the value of “a” will be the GCD of the original two numbers.

This algorithm is NOT intuitive! Your task is not to decipher why it works, but to implement the algorithm as defined above. I.e. you are learning to map English into java.

To program this algorithm, create a new text document “EuclidGCD.java” in the folder “lab04” and write a “EuclidGCD” class with a “main” method in it. This method should read two integer variables from the user, and will be the initial numbers from which the GCD will be calculated. Note that any integer can be given as input, such as positive or negative numbers, and that either could be greater than the other. Based on these numbers you will need to:

• discover whether both are positive numbers; otherwise a message saying that the program only works with numbers greater than 0 is displayed and no computation is attempted. The message should say “POSITIVE NUMBERS ONLY”.
• discover which one is “a” or “b”, where “a” is greater than “b”; if a is less than b, swap the values.
• print the GCD resulting from the algorithm above.

If you are not sure whether to use a “for” or “while”, ask your instructor.

The figure below shows the case in which one of the numbers is not a positive number.
The figure below shows the case in which valid numbers are provided.

![Command Prompt Window](image)

**Testing**

Test that your program calculates areas correctly by using the following values:

- input = 15, 5; GCD = 5
- input = 12, 18; GCD = 6
- input = 42, 56; GCD = 14
- input = 126, 84; GCD = 42
- input = 72390, 15000; GCD = 30

When you are finished, submit your program to Web-CAT.
Lab 4: Completion Table

```java
int i=1;
while (i<=6)
{
    System.out.println("Try Again");
    i++;
}
```

Be sure to submit Labs 4A, 4B, and 4C to Web-CAT!!!